論　文

ロバストネス性を考慮した容量制約をもつネットワーク設計問題

片山　直登

1　はじめに

ネットワーク設計問題は，ノードおよび容量をもつアーク候補からなるネットワークと，ネットワーク上を流れる多品種の需要量が与えられたときに，ネットワークのデザイン費用とフロー費用の合計が最小となるアークの選択と各品種のフローの経路を求める問題である．これは，通信ネットワークの設計や輸送・配送ネットワークの設計など，幅広い分野で応用されている問題（Magnanti et al. 1986）であり，多くのサーベイ（Balakrishnan et al. 1997, Costa 2005, Crainic 2003, Gendron et al. 1997, Magnanti and Wong 1984, Minoux 1989, Wong 1984, 1985, Yaghini and Rahbar 2012）がまとめられている．また，この問題は NP- 困難な問題であることが知られている（Magnanti and Wong 1984）．

一般的な容量制約をもつネットワーク設計問題は，各品種の需要が既知として与えられている．しかし，現実的には，需要は不確実なものである場合が多い．このような需要の不確実性を取り扱った問題は，確率的な容量制

約をもつネットワーク設計問題やロバストネス性を考慮した容量制約をもつネットワーク設計問題とよばれている．ロバストネス性は，需要の変動に対するネットワークの頑強性，すなわち様々な需要変動に対して適切に処理できる設計を行うことを意味している．ロバストネス性を考慮した容量制約をもつネットワーク設計問題では，多くのシナリオを設定し，これらのシナリオに対して，平均的に優れたネットワーク設計を行うシナリオ解析が行われている．シナリオ解析は，不確実性要因に対処するための複数の異なる条件を設定したインスタンスであるシナリオを用意し，これらのシナリオに対して解析を行う手法である．このような問題を，シナリオ解析を用いたロバストネス性を考慮した容量制約をもつネットワーク設計問題（*SCND*）とよぶ．

これまでに数多くのロバストネスあるいは確率的ネットワーク設計問題の研究がなされており，サービスネットワーク設計問題に対するネットワーク構造とネットワークフローの 2 段階モデル（Lium et al. 2009, Crainic et al. 2011）や不確実需要下における多段階エシュロン在庫を含むサプライチェーンネットワーク問題に対する研究（A. Alonso-Ayuso et al. 2003, Bidhandi and Yusuff 2011）等がある．ロバストネス性や確率的ネットワーク設計問題に対して，シナリオを用いた解析（Tsiakis et al. 2001, Hoyland and Wallace 2001, Smith et al. 2004）やサンプリングに基づいた静的シナリオを用いた標本平均近似法を用いた研究（Santoso et al. 2005，Azaron et al. 2008，Schütz et al. 2009）がある．

静的なシナリオが与えられた場合，ネットワーク設計問題はこれらのシナリオを含む大規模な静的な最適化問題として捉えることができる．近年では，CPLEX や Gurobi といった商用の最適化ソルバーが改良され，ある程度の規模の最適化問題を高速に最適に解くことができるようになってきている．しかしながら，多数のシナリオを含む大規模なインスタンスの場合は，直接的に最適化ソルバーで解くことは困難である．このため，

Lagrange 緩和等を用いて全体の問題をシナリオ毎の問題に分解することによる，分枝固定法やプログレッシブヘッジに基づいたメタヒューリスティクス解法（Schütz et al. 2009, Crainic et al. 2011, 2014）が示されている．

一方，メタヒューリスティクスに代表されるネットワーク設計問題に対する近年の研究では，精度の高い解の算出が議論の中心となっている．一般に，メタヒューリスティクス解法では，精度の高い解を算出するためには多くの計算時間を必要としている．しかし，実用的な問題ではノード数，アーク数や品種数が膨大なものとなることから，適度な精度の解を算出する高速な解法の開発が必要となっている．さらには，様々なシナリオに柔軟に対応できるネットワーク設計を目的とするロバスト性を考慮したネットワーク設計問題や不確実性を考慮したネットワーク設計問題では，多くのシナリオやデータに対する膨大なネットワーク設計問題を繰り返し解くことが必要となることから，大規模なネットワーク設計問題を短時間で解を算出できる高速な解法の開発が望まれている．容量制約のないネットワーク設計問題に対する高速解法として，貪欲法であるデリート法が知られている．Minoux（1989）はこの貪欲法を改良した解法を示し，片山（2010）は Minoux の貪欲法を改良した解法を提案している．片山（2015）は容量制約をもつネットワーク設計問題に貪欲法を応用し，容量スケーリング法を用いることによって早期にアーク候補を絞り，貪欲法と最適化ソルバーを組み合わせたことにより，高速でかつ精度の高い解を算出できる解法を提案している．

本研究では，需要が不確実，すなわち需要のシナリオとその発生確率が与えられたもとで，ロバストネス性を考慮した容量制約のあるネットワーク設計問題 *SCND* を取り扱う．*SCND* に対して高速な貪欲法を用いた近似解法を提案し，さらには容量スケーリング法や限定した分枝限定法を組合せた高速な近似解法を提案する．

2　*SCND* の定式化

ノード集合を N，アーク候補集合を A，品種の集合を K とする．アーク (i, j) を選択するか否かを表すデザイン変数を y_{ij} とし，アーク (i, j) の容量を C_{ij}，デザイン費用を f_{ij} とし，アーク (i, j) 上の品種 k の単位当たりのフロー費用を c_{ij}^{k} とする．また，パス p がアーク (i, j) を含むとき 1，そうでないとき 0 である定数を δ_{ij}^{p} とする．続いて，対象とするシナリオ集合を S とし，シナリオ s の発生確率を e^{s} とする．シナリオ s における品種 k の取り得るパス集合を P^{ks} とし，全体のパス集合を P とする．また，シナリオ s に対する品種 k のパス p のフロー量であるシナリオパスフロー変数を x_{p}^{ks} とし，シナリオ s における品種 k の需要量を d^{ks} とする．

このとき，アーク集合 A，パス集合 P，シナリオ集合 S に対して，シナリオパスフロー変数を用いた *SCND* の定式化 *SCNDP*（A, P, S）は，次のように表される．

（*SCNDP*（A, P, S））

$$\min \sum_{(i,j)\in A} f_{ij} y_{ij} + e^{s} \sum_{s\in S} \sum_{(i,j)\in A} \sum_{k\in K} c_{ij}^{ks} \sum_{p\in P^{ks}} \delta_{ij}^{p} x_{p}^{ks} \tag{1}$$

subject to

$$\sum_{p\in P^{ks}} x_{p}^{ks} = d^{ks} \qquad \forall k\in K, s\in S \tag{2}$$

$$\sum_{k\in K} \sum_{p\in P^{ks}} \delta_{ij}^{p} x_{p}^{ks} \leq C_{ij} y_{ij} \qquad \forall (i,j)\in A, s\in S \tag{3}$$

$$\sum_{p\in P^{ks}} \delta_{ij}^{p} x_{p}^{ks} \leq d^{ks} y_{ij} \qquad \forall (i,j)\in A, k\in K, s\in S \tag{4}$$

$$x_p^{ks} \geq 0 \quad \forall p \in P^{ks},\ k \in K,\ s \in S \tag{5}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{6}$$

(1)式は，デザイン費用とフロー費用の期待値の和であり，これを最小化することを表す．フロー費用は，各シナリオの発生確率とフロー費用の積の和である期待値として捉えている．(2)式は，シナリオ s の品種 k のシナリオパスフロー量の和はシナリオ s における需要量になることを表す需要保存式である．(3)式の左辺はシナリオ s におけるアーク (i, j) 上のフロー量の合計であり，右辺はアーク (i, j) が選択されるときに C_{ij}，選択されないとき0となるシナリオとアークごとの容量制約式である．(4)式の左辺はシナリオ s におけるアーク (i, j) 上の品種 k のシナリオパスフロー量の合計であり，右辺はアーク (i, j) が選択されるときに d^{ks}，選択されないとき0となるシナリオ，品種とアークごとの強制制約式である．(5)式はシナリオパスフローの非負制約，(6)式はデザイン変数の0−1条件である．

SCND には，2種類の変数，デザイン変数 y とシナリオフロー変数 x が含まれる．そこで，デザイン変数 y が1となるアークを選択する問題，すなわち全体のアーク候補集合からアーク集合を選択するネットワーク構造に関する問題と，ネットワーク構造が確定したネットワーク上でフローを求める多品種フロー問題の2段階の問題として，*SCND* を考える．

すべてのデザイン変数が固定された *SCND* を考える．このとき，$y_{ij}=1$ である選択されたアーク集合を $\overline{A}$ とする．選択されたアーク集合が $\overline{A}$ に確定した問題は，シナリオごとの問題に分離することができ，それぞれの問題は $\overline{A}$ からなるネットワーク上の多品種フロー問題となり，これらを解くことにより各シナリオに対するフロー費用を求めることができる．

ここで，アーク集合 A の部分集合 $\overline{A}$ に対するシナリオ s の多品種フロー

問題を $MCF^s(\overline{A})$ とし，$MCF^s(\overline{A})$ の最適目的関数値であるシナリオ s におけるフロー費用を $\phi^s(\overline{A})$ とおく．シナリオ s におけるアーク (i, j) 上の品種 k のフロー量を表すシナリオアークフロー変数を x_{ij}^{ks} とし，品種 k の始点を O^{ks}，終点を D^{ks} とする．また，$N_n^+(\overline{A})$ はアーク $\overline{A}$ からなるネットワーク上でノード n から出るアーク集合，$N_n^-(\overline{A})$ はアーク $\overline{A}$ からなるネットワーク上でノード n に入るアーク集合である．

シナリオアークフロー変数を用いた $SCND$ の定式化 $SCNDA$（A, S）は，次のように表すことができる．

（$SCNDA(A.\ S)$）

$$\min_{\forall \overline{A} \subseteq A} \sum_{(i,j)\in\overline{A}} f_{ij} + e^s \sum_{s\in S} \phi^s(\overline{A}) \tag{7}$$

subject to

（$MCF^s(\overline{A})$）, $s \in S$）

$$\phi^s(\overline{A}) = \min \sum_{(i,j)\in\overline{A}} \sum_{k\in K} c_{ij}^k x_{ij}^{ks} \tag{8}$$

subject to

$$\sum_{i\in N_n^+(\overline{A})} x_{in}^{ks} - \sum_{i\in N_n^-(\overline{A})} x_{nj}^{ks} = \begin{cases} -d^{ks} & if\ n = O^{ks} \\ d^{ks} & if\ n = D^{ks} \\ 0 & ohterwise \end{cases} \qquad \forall n \in N,\ k \in K \tag{9}$$

$$\sum_{k\in K} x_{ij}^{ks} \le C_{ij} \qquad \forall (i,j) \in \overline{A} \tag{10}$$

$$0 \le x_{ij}^{ks} \le d^{ks} \qquad \forall k \in K,\ (i,j) \in \overline{A} \tag{11}$$

(7)式は，選択されたアーク集合 $\overline{A}$ に含まれるアークのデザイン費用と $\overline{A}$ に対するシナリオ s のフロー費用の期待値の和であり，これを最小化することを表す．(8)式は，選択されたアーク集合 $\overline{A}$ に対するシナリオごと

の総フロー費用であり，これを最小化することを表す．(9)式は，ノード n における流入量と流出量の差が，ノード n が品種 k の始点 O^{ks} であれば $-d^{ks}$，終点 D^{ks} であれば d^{ks}，その他のノードであれば 0 となることを表すフロー保存式である．(10)式の左辺はアーク (i, j) 上のフロー量の合計であり，これが容量 C_{ij} 以下となる容量制約式である．(11)式はシナリオアークフロー変数の上限と非負制約である．

3　貪欲法

容量制約をもつネットワーク設計問題に対して，片山（2015）が高速な貪欲法を提案している．ここでは，シナリオアークフローによる定式化 $SCNDA(A, S)$ に対してこの貪欲法を適用する．

この貪欲法は，最も目的関数値が減少するようなアークを順次削除するデリート法を基本としている．選択するアークを決めると，$SCNDA(A, S)$ はシナリオごとの多品種フロー問題に分割できる．始めに，すべてのネットワーク上のアークに対して，それぞれのアークを削除したネットワークにおけるシナリオごとの多品種フロー問題 $MCF^s(A)$ を解き，これらからアークを削除したときの目的関数値の減少量の期待値を求める．続いて，目的関数値の減少量の期待値が最大であるアークを削除する操作を繰り返す．

1 本のアークを削除するとネットワークの構造が変わるため，本来であれば残っているすべてのアークに対して，このアークを削除したネットワークにおけるシナリオごとの多品種フロー問題を再度解き，アークを削除したときの目的関数値の減少量の期待値を求め直すことが必要となる．しかし，これでは計算量が膨大なものとなる．そこで，各アークを削除したときの目的関数値の減少量の期待値は，現時点で減少量が最大ではない

アークでは前回の目的関数値の減少量の期待値をそのまま利用し，減少量が最大のアークについてのみ減少量の期待値を厳密に再計算する．この値が依然アークの中で最大であればこのアークを削除し，そうでなければこのアークの減少量の期待値の更新だけを行う．この方法により，目的関数値の減少量の期待値の算出回数を大幅に削減することができる．

貪欲法のアルゴリズムを Algorithm1に示す．ψ_{ij} はアーク（i, j）を削除したときの目的関数値の減少量の期待値であり，L は減少量の期待値が正であるアークの集合である．また，π は L に含まれるアークの中で ψ_{ij} の最大値，（i^*, j^*）は ψ_{ij} が最大であるアークである．アーク（i^*, j^*）を L から取り出した後，アーク（i^*, j^*）について $\psi_{i^*j^*}$ を再計算する．この値が L に含まれるアークの減少量の期待値の中で最大値以上であればアーク（i^*, j^*）を削除し，そうでなければ L に戻す．

このアルゴリズムでは，アーク集合 A における目的関数値の減少量の初期の計算回数は $O(|A|)$ であり，それ以降はアーク（i, j）が実際に削除の対象となる場合に限り目的関数値の減少量を計算するだけである．このため，多品種フロー問題を解く回数は最悪の場合には $O(|A|^2)$ であるが，経験的には $O(|A|)$ となる．

4　容量スケーリング法と限定した分枝限定法

前節で示した貪欲法は，効率的に近似解を算出できる可能性がある．この解法では，ネットワークから特定のアークを削除したときの目的関数値の減少量を基準とし，減少量の大きい順にアークを削除していくことが手順となる．大規模なネットワーク上で品種数や需要量の少ない問題に対して，アーク集合 A からなるネットワーク上で多品種フロー問題を解いた場合には，大半のアーク上のフロー量は 0 となることが予想される．フ

ロー量が0であるアークを削除しても，目的関数の減少量はそのアークのデザイン費用のみであることから，フロー量が0でかつデザイン費用の高いアークから優先的に削除される可能性がある．このため，これらのアークが多数最適解に含まれる場合では，得られる近似解の精度が大きく悪化する可能性が高くなる．一方，貪欲法が効率的であるとは言っても，アーク数の多い問題では非常に多くの多品種フロー問題を繰り返し解く必要がある．これらの問題を解決するために，初期ネットワークに対して容量スケーリング法を適用し，事前に近似解に含まれる可能性の高いアークに絞ることにする．

容量スケーリング法は，線形緩和問題を解き，そのデザイン変数解の値に従ってアーク容量を変化させ，0または1のデザイン変数解を導出するものである．容量スケーリングを行うことによりアーク容量が減少するために，多数のアーク上にフローが分散することになる．また，少ない繰り返し回数で多くのデザイン変数が0に収束することも知られている（Katayama 2015）．そこで，アーク集合Aに対して容量スケーリング法を適用し，0に収束しないデザイン変数のみを選定し，これらのアークのみを貪欲法の対象にする．この前処理により，わずかな計算量で多くのアークを除外することができることになり，効率的に問題規模を縮小することが可能となる．もちろん，0に収束したデザイン変数が最適解において1である可能性があり，最適解を排除する可能性があることに注意する．なお，すべての変数が0または1に収束するためには多くの計算時間が必要なため，0に収束しないデザイン変数が一定数以下となったら容量スケーリングを終了し，これらのデザイン変数に対応するアークを貪欲法の対象となるアーク候補集合に限定する．

容量スケーリング法では，定式化$SCNDP(A, P, S)$を利用する．$SCNDP(A, P, S)$において，(3)式にあるアーク容量をC_{ij}^{l}とし，(6)式の

上限である1を C_{ij}/C_{ij}^{l}，0を下限に置き換えた線形緩和問題を $SCNDPL(A, P, S, C^{l})$ とする．なお，l は容量スケーリングの繰り返し回数である．

($SCNDPL(A, P, S, C^{l})$)

$$\min \ \min \sum_{(i,j)\in A} f_{ij}y_{ij} + e^{s}\sum_{s\in S}\sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^{ks}\sum_{p\in P^{ks}}\delta_{ij}^{p}x_{p}^{ks} \tag{12}$$

subject to

$$\sum_{p\in P^{ks}} x_{p}^{ks} = d^{ks} \qquad \forall k\in K, s\in S \tag{13}$$

$$\sum_{k\in K}\sum_{p\in P^{ks}}\delta_{ij}^{p}x_{p}^{ks} \le C_{ij}^{l}y_{ij} \qquad \forall (i,j)\in A, s\in S \tag{14}$$

$$\sum_{p\in P^{ks}}\delta_{ij}^{p}x_{p}^{ks} \le d^{ks}y_{ij} \qquad \forall (i,j)\in A, k\in K, s\in S \tag{15}$$

$$x_{p}^{ks} \ge 0 \qquad \forall p\in P^{ks}, k\in K, s\in S \tag{16}$$

$$y_{ij} \le C_{ij}/C_{ij}^{l} \qquad \forall (i,j)\in A \tag{17}$$

$l-1$ 回目の繰り返しである $SCNDPL$（A, P, S, C^{l-1}）のデザイン変数解を $\tilde{y}$ とすると，l 回目のアーク容量を次のように変更する．

$$C_{ij}^{l} := \lambda C_{ij}^{l-1}\tilde{y}_{ij} + (1-\lambda)C_{ij}^{l-1} \tag{18}$$

ここで，はスケーリングパラメータであり，0から1の間の定数である．

$SCNDP(A, P, S)$ や $SCNDA(A, S)$ は組合せ最適化問題であるため，大規模な問題を最適に解くことは困難である．一方，$SCNDPL(A, P, S, C^{l})$ はパスやアークが限定された線形計画問題であるため，ある程度の規模の問題であればMIPソルバーを用いて解くことができる．

容量スケーリング法のアルゴリズムをAlgorithm2に示す．ε は収束判

定基準，ITE_{min} は容量スケールの最小繰り返し回数，ITE_{max} は容量スケールの最大繰り返し回数である．また，$ArcNum$ は収束していないアーク数の上限値であり，収束していないアーク数が $ArcNum$ 以下となった場合，容量スケーリングを終了する．なお，パスフローを用いた定式化では，必要なパスを生成する列生成法と必要な強制制約式を生成する行生成法を用いる．容量スケーリング法ならび列生成法と行生成法の詳細については，Katayama et al.（2009）を参照のこと．

容量スケーリング法を適用した結果，容量スケーリングをしたネットワーク上におけるパスフロー変数および対応する強制制約式が生成される．生成されたパスの集合を $\tilde{P}(\in P)$ とおき，0 に収束しないデザイン変数に対するアーク集合を $\tilde{A}$ とする．このとき，アーク集合 $\tilde{A}$ とパス集合 $\tilde{P}$ で構成される問題は $SCNDP(\tilde{A}, \tilde{P}, S)$ となる．$\tilde{A}$ と $\tilde{P}$ はそれぞれ A と P の部分集合であるため，$SCNDP(\tilde{A}, \tilde{P}, S)$ の最適解は $SCNDP(A, P, S)$ の近似解，最適値は $SCNDP(\tilde{A}, \tilde{P}, S)$ の上界値となる．

$SCNDP(\tilde{A}, \tilde{P}, S)$ を最適または近似的に解くことによって，$SCNDA(A, S)$ の近似解を求めることができる．本来の $SCNDA(A, S)$ に比べると，$SCNDP(\tilde{A}, \tilde{P}, S)$ の規模は相対的に小さいため，最適化ソルバーの分枝限定法を用いると短時間で $SCNDP(\tilde{A}, \tilde{P}, S)$ の最適解を算出できる可能性がある．もちろん，短時間で最適解を算出できる保証はないため，計算時間の上限値を設定して，$SCNDP(\tilde{A}, \tilde{P}, S)$ を解くことにする．この方法を限定した分枝限定法とよぶ．

$SCNDP(\tilde{A}, \tilde{P}, S)$ はパス変数が限定されているため，$\tilde{A}$ に対する厳密な問題 $SCNDP(\tilde{A}, P, S)$ と比べると，最適解に含まれるすべてのパスが含まれていない可能性があり，$SCNDP(\tilde{A}, \tilde{P}, S)$ の目的関数値は $SCNDP(\tilde{A}, P, S)$ の目的関数値よりも大きな値となる．そこで，$SCNDP(\tilde{A}, \tilde{P}, S)$ で得られたアーク集合 $\tilde{A}$ を用いた $MCF^s(\tilde{A})$ を解くことによって $\phi^s(\tilde{A})$

を求め，$SCNDA(A, S)$ の目的関数値を算出する．これは，$SCNDP(\bar{A}, \bar{P}, S)$ ではパス変数が限定されており，$MCF^s(\bar{A})$ を用いた方がより良い目的関数値を得られる可能性があるためである．

また，スケーリングを行う以前の容量が $C^l = C$ である線形緩和解において，正となるデザイン変数の数が少数で $ArcNum$ 以下である場合には，これらのデザイン変数をもつアークも貪欲法のための事前に限定したアーク候補集合に含め，限定分枝限定法の対象となるアークの範囲を増加させることにする．

全体のアルゴリズムを Algorithm3に示す．始めに容量スケーリング法でパスを生成しかつ対象アークを限定し，容量スケーリング法により限定されたアーク集合とパス集合を用いて限定された分枝限定法により近似解を算出する．続いて，貪欲法を用いて近似解を算出する．なお，容量スケーリング法および限定された分枝限定法にはパスフローを用いた定式化 $SCNDP$ を用い，貪欲法にはアークフローを用いた定式化 $SCNPA$ を用いることに注意する．アークフローを用いた定式化 $SCNPA$ を用いるのは，フロー問題のみを解く場合には，アークフローを用いた定式化がより高速に解けるためである．

5　おわりに

本研究では，需要が不確実，すなわち需要のシナリオとその発生確率が与えられたもとで，ロバストネス性を考慮した容量制約のあるネットワーク設計問題 $SCND$ に対して，高速な貪欲法を提案し，さらに容量スケーリング法ならびに限定された分枝限定法を組合せた解法を提案した．今後の課題として，ベンチマーク問題を用いた数値実験による従来の解法との比較や現実問題への適用が挙げられる．

なお，本研究は科学研究費基盤研究Ｃ（課題番号25350454）による成果の一部である．

参考文献

A. Alonso-Ayuso, L.F. Escudero, A. Garín, M.T. Ortuño and G. Pérez. An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26: 97－124, 2003.

A. Azaron, K.N. Brown, S.A. Tarim, and M. Modarres. A multi-objective stochastic programming approach for supply chain design considering risk. *International Journal of Production Economics*, 116: 129－138, 2008.

A. Balakrishnan, T.L. Magnanti, and P. Mirchandani. Network design. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 311－334. John Wiley & Sons, New York, 1997.

H.M. Bidhandi and R.M. Yusuff. Integrated supply chain planning under uncertainty using an improved stochastic approach. *Applied Mathematical Modelling*, 35: 2618－2630, 2011.

A.M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research*, 32: 1429－1450, 2005.

T.G. Crainic. Long-haul freight transportation. In R.W. Hall, editor, *Handbook of Transportation Science*, pages 451－516. Kluwer Academic Publishers, 2003.

T.G. Crainic, M. Hewitt, and W. Reia. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Computers & Operations Research*, 43: 90－99, 2014.

T.G. Crainic, Fu X, M. Gendreau, W. Rei, and S.Wallace. Progressive hedging-based meta-heuristics for stochastic network design. *Networks*, 58: 114－124, 2011.

B. Gendron, T.G. Crainic, and A. Frangioni. Multicommodity capacitated network design. Technical Report CIRRELT-98-14, Centre de recherche sur les transports, Universite de Montreal, 1997.

K. Hoyland and S.W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47: 295－307, 2001.

N. Katayama. A combined capacity scaling and local branching approach for capacitated multi-commodity network design problem. *Far East Journal of Applied Mathematics*, 92: 1－30, 2015.

N. Katayama, M.Z. Chen, and M. Kubo. A capacity scaling procedure for the multi-commodity capacitated network design problem. *Journal of Computational and Applied Mathematics*, 232: 90－101, 2009.

A. Lium, T.G. Crainic, and S.W. Wallace. A study of demand stochasticity in service network design. *Transportation Science*, 43: 144－157, 2009.

T.L. Magnanti and R.T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18: 1－55, 1984.

T.L. Magnanti, P. Mireault, and R.T.Wong. Tailoring benders decomposition for uncapacitated network design. *Mathematical Programming Study*, 26: 112－155, 1986.

M. Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, 19: 313－360, 1989.

T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167: 96－115, 2005.

P. Schütz, A. Tomasgard, and S. Ahmed. Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, 199: 409－419, 2009.

J.C. Smith, A.J. Schaefer, and J.W. Yen. A stochastic integer programming approach to solving a synchronous optical network ring design problem. *Networks*, 44: 12－26, 2004.

P. Tsiakis, N. Shah, and C.C. Pantelides. Design of multi-echelon supply chain networks under demand uncertainty. *Industrial & Engineering Chemistry Research*, 40: 3585－3604, 2001.

R.T. Wong. Introduction and recent advances in network design: Models and algorithms. In M. Florian, editor, *Transportation Planning Models*, pages 187－225. Elsevier Science, North Holland, Amsterdam, 1984.

R.T. Wong. Location and network design. In M. O'heEigeartaigh, J. Lenstra, and A. Rin-nooyKan, editors, *Combinatorial Optimization Annotated Bibliographies*, pages 129－147. John Wiley & Sons, New York, 1985.

M. Yaghini and M. Rahbar. Multicommodity network design problem in rail freight transportation planning. *Procedia-Social and Behavioral Sciences*, 43: 728－739, 2012.

片山直登．多品種を考慮したロジスティクスネットワーク設計問題の数理的解法に関する研究．博士論文，流通経済大学，2010.

片山直登. 容量制約をもつネットワークデザイン問題の高速な貪欲解法．流通情報学部紀要，20（1）: 1－24, 2015.

Algorithm 1: Greedy Algorithm(A)

$L \leftarrow ()$;
$\psi_{ij} \leftarrow 0$;
for $s \in S$ **do**
  Solve $MCF^s(A)$;
  Get $\phi^s(A)$;
  for $(i,j) \in A$ **do**
    Solve $MCF^s(A \backslash \{(i,j)\})$;
    if $MCF^s(A \backslash \{(i,j)\})$ *is feasible* **then**
      Get $\phi^s(A \backslash \{(i,j)\})$;
      $\psi_{ij} \leftarrow \psi_{ij} + e^s\{\phi^s(\bar{A}) - \phi^s(A \backslash \{(i,j)\})\}$;
    else
      $\psi_{ij} \leftarrow 0$;
      break;
  end
end
for $(i,j) \in A$ **do**
  if $\psi_{ij} > 0$ **then** $L.push((i,j))$;
end
$\bar{A} \leftarrow A$;
if $|L| > 0$ **then** $\pi \leftarrow \max_{(i,j)\in L} \psi_{ij}$;
while $|L| > 0$ *and* $\pi > 0$ **do**
  $(i^*, j^*) \leftarrow \arg\max_{(i,j)\in L} \psi_{ij}$;
  $L.pop((i^*, j^*))$;
  $\psi_{i^*,j^*} \leftarrow 0$;
  for $s \in S$ **do**
    Solve $MCF^s(\bar{A} \backslash \{(i^*, j^*)\})$;
    if $MCF^s(\bar{A} \backslash \{(i^*, j^*)\})$ *is feasible* **then**
      Get $\phi^s(\bar{A} \backslash \{(i^*, j^*)\})$;
      $\psi_{i^*j^*} \leftarrow \psi_{i^*j^*} + e^s\{\phi^s(\bar{A}) - \phi^s(\bar{A} \backslash \{(i^*, j^*)\})\}$;
    else
      $\psi_{i^*j^*} \leftarrow 0$;
      break;
  end
  if $\psi_{i^*j^*} \geq \pi$ **then**
    $\bar{A} \leftarrow \bar{A} \backslash \{(i,j)\}$;
  else
    $L.push((i^*, j^*))$;
  if $|L| > 0$ **then** $\pi \leftarrow \max_{(i,j)\in L} \psi_{ij}$;
end
Get $\phi(\bar{A})$;
$UB_{MG} \leftarrow \phi(\bar{A})$;
Return $\bar{A}, UB_{MG}$;

Algorithm 2: Capacity Scaling(A)

Set $\bar{P}$, λ, ϵ, ITE_{min}, ITE_{max}, $ArcNum$;
Solve *SCNDPL(A, P, S, C)*;
Get the solution $\tilde{y}$ of *SCNDPL(A, P, S, C)*;
Add paths to $\bar{P}$ by Column Genaration;
$\hat{A} \leftarrow ()$;
Get AN which is the number of arcs such that $y_{ij} > \epsilon$;
if $AN < ArcNum$ **then**
  for $(i, j) \in A$ **do**
    if $\tilde{y}_{ij} > \epsilon$ **then**
      $\hat{A}.push((i, j))$;
    end
  end
end
$C^1 \leftarrow C$; $l \leftarrow 1$;
repeat
  Solve *SCNDPL(A, P, S, C^l)*;
  Get the solution $\tilde{y}$ of *SCNDPL(A, P, S, C^l)*;
  Add paths to $\bar{P}$ by Column Genaration;
  $\bar{A} \leftarrow ()$;
  for $(i, j) \in A$ **do**
    $C_{ij}^l \leftarrow \lambda C_{ij}^{l-1} \tilde{y}_{ij} + (1 - \lambda) C_{ij}^{l-1}$;
    if $\tilde{y}_{ij} > \epsilon$ **then**
      $\bar{A}.push((i, j))$;
      $\hat{A}.push((i, j))$;
    end
  end
until $l \geq ITE_{min}$ *and* $|\bar{A}| \leq ArcNum$, *or* $l \geq ITE_{max}$
Return $\bar{A}$,$\hat{A}$,$\bar{P}$;

Algorithm 3: Combined Algorithm

Set A;
$\bar{A}$,$\hat{A}$,$\bar{P} \leftarrow$ Capacity Scaling(A);
Solve *SCNDPL($\hat{A}$, $\bar{P}$, S)*; and get $\tilde{A}$ which is the selected arc set;
Get $\phi(\tilde{A})$;
$UB_{BB} \leftarrow \phi(\tilde{A})$;
$\bar{A}, UB_{MG} \leftarrow$ Greedy Algorithm($\bar{A}$);
$UB \leftarrow \min(UB_{BB}, UB_{MG})$;
Return $\bar{A}, UB$;